PYTHON PROGRAMLAMA

Ders 9: Veritabanı İşlemleri

Öğr. Gör. Dr. Murat ASLANYÜREK

Kırklareli Üniversitesi
Pınarhisar MYO

Veri Tabanları

- Geçen hafta belirttiğimiz gibi; kayıt tabanlı işlemler için günümüzde çoğunlukla veritabanları kullanıldığından rastgele erişimli dosyalama işlemlerine ihtiyaç kalmamıştır.
- Bir personelin adı, soyadı, doğum tarihi, maaşı gibi veya internetten yapılan bir alışverişin tarihi, kimin tarafından yapıldığı, hangi adrese gönderileceği gibi bilgiler bir kayıt olarak veritabanında saklanır.
- Bu tip kayıtları saklamak için genellikle ilişkisel veri tabanları tercih edilir ve bu veritabanlarında her kayıt ilgili tabloda bir satırı oluşturur. Ad ve soyad gibi nitelikler ise sütunları oluşturur.

Temel SQL Komutları

- CREATE: Veritabanında bir nesne yaratmak için kullanılır. Bu derste sadece CREATE TABLE ile tablo nesnesi yaratma gösterilecektir.
- DROP: Veritabanından nesne silme için kullanılır.
  - DROP TABLE Personel komutu Personel tablosunu siler.
- ALTER: Veritabanındaki nesneler üzerinde değişiklik yapmak için kullanılır (Örneğin tabloya yeni sütun ekleme).
- INSERT: Tabloya kayıt (satır) ekleme için kullanılır.
- DELETE: Tablodan kayıt silmek için kullanılır.
- UPDATE: Tablodaki kayıtları değiştirmek için kullanılır.
- SELECT: Tablodaki kayıtları elde etme (sorgu yapma) için kullanılır.

CREATE TABLE ile tablo yaratma

- Veritabanında yeni bir tablo yaratmak için aşağıdaki gibi bir komut kullanabiliriz:

```
CREATE TABLE Personel(
    PerNo integer primary key,
    Ad char(10),
    Soyad char(10),
    Maas float)
```

"Primary Key" tanımı o alanın birincil anahtar olduğunu, yani tüm kayıtlarda farklı değere sahip olması gerekeceğini gösterir.

char(10) = 10 karakterlik string

- Bu komut, veritabanında PerNo, Ad, Soyad ve Maas alanlarına (sütunlarına) sahip Personel adında bir tablo yaratacaktır.
- NOT: Veri Tabanı Yönetim Sistemlerinde (VTYS) tablolar aslında birer nesne olarak değerlendirilse de, programlama dillerindeki sınıfa benzer bir yapıdadırlar. NYP yaklaşımı ile ilişkilendirecek olursak; Personel bir sınıfa, içindeki 4 alan o sınıfın niteliklerine ve tabloya eklenen her kayıt ise o sınıftan üretilen nesnelere denk gelir.

SQLite veritabanında tablo yaratılırken integer, char(10) gibi veri türlerini yazma zorunluluğu yoktur. Verilmediği durumda veri türü sınırsız bir string olan TEXT olarak kabul edilir. Fakat çoğu ilişkisel VT'de tablo yaratılırken veri türleri mutlaka belirtilmelidir.

INSERT ile kayıt ekleme

- Bir tabloya kayıt eklemek için INSERT INTO ifadesi kullanılır. Bu ifadeden sonra tablonun ismi, ardından VALUES ile kayıttaki verileri içeren bir tuple yazılır:

```
INSERT INTO Personel
VALUES(4, 'Altan', 'Mesut', 1234.45)
```

- Eğer tablodaki sadece belli alanlara veri göndereceksenız o alanların isimlerini içeren bir tuple da tablo isminden sonra vermelisiniz:

```
INSERT INTO Personel(PerNo, Ad, Soyad)
VALUES(4, 'Altan', 'Mesut')
```

ALTER TABLE ile nitelik ekleme

- Diyelim ki daha önce yarattığımız bir tabloya yeni bir nitelik (sütun) eklemek istedik.
  - Eğer DROP ile tabloyu yok ederek tekrar yaratırsak içindeki tüm verileri kaybetmiş oluruz.
  - Eğer ALTER ile bir nitelik eklersek, mevcut kayıtlarda bu yeni nitelik alanı boş (None - Null) olacak şekilde mevcut kayıtlar silinmeden bu işlemi gerçekleştirebiliriz.
  - Personel tablosuna Görev niteliğini aşağıdaki komutla ekleyebiliriz:

```
ALTER TABLE Personel ADD Görev
```

SELECT ifadesi

SELECT sütun [yada sütunlar]

FROM tablo [yada tablolar]

WHERE seçim kriteri

- SELECT ifadesinden sonra * kullanılırsa tüm nitelikler (sütunlar) seçilir.
- Kriter verilmezse “WHERE” sözcüğü de yazılmaz. Bu durumda tüm kayıtlar (satırlar) seçilir.
- Personel tablosundaki tüm kayıtların tüm nitelikleri aşağıdaki SQL cümlesi ile gösterilir:

```
SELECT * FROM Personel
```

WHERE ile kriter verme

- Eğer tüm kayıtların değil de sadece belirli kayıtların görüntülenmesi istenirse WHERE ile kriter verilir.
- Aşağıdaki SQL cümlesi maaşı 1800 TL'nin üzerinde olan personelin adı ve soyadını ekranda gösterir:

```
SELECT Ad, Soyad
FROM Personel
WHERE Maas > 1800
```

Kriterlerde kullanılan işleçler

- Programlama dillerinde kullanılan aritmetiksel karşılaştırma işleçleri (<, <=, >, >=, =, <>) ve mantıksal işleçler (AND, OR, NOT) SQL dilinde de kriter verirken kullanılır.
- Aşağıdaki SQL cümlesi görevi müdür olan ve maaşı 5000 TL'den fazla olan personeli gösterir:

```
SELECT * FROM Personel
WHERE Görev = 'Müdür'
   AND Maas > 5000
```

Programlama dillerinde olduğu gibi SQL'de de karakter türü veriler ile işlem yapılacaksa tek tırnak yada çift tırnak kullanılır.

UPDATE ile kaydı güncelleme

- Eğer bir veya daha çok kayıtta yer alan bir bilgiyi değiştirmek istersek UPDATE komutunu kullanabiliriz.
- Hangi kayıtların değiştirileceğini belirtmek için yine WHERE ile bir kriter verebilir ve hangi niteliğe hangi değişikliğin yapılacağını SET ile belirtiriz.
- Aşağıdaki ifade Personel tablosundaki Görev niteliği sekreter olan kayıtların Maas niteliğine %15 zam ekler:

```
UPDATE Personel
SET Maas = Maas * 1.15
WHERE Görev = 'Sekreter'
```

Eğer WHERE ile kriter verilmezse, tüm kayıtlar için SET ile belirtilen değişiklik yapılır.

DELETE ile kayıt silme

- Tablodan kayıt silmek için DELETE FROM ifadesi kullanılır.
- Hangi kayıtların silineceği yine WHERE ile belirtilebilir:

```
DELETE FROM Personel
WHERE PerNo = 35
```

  - PerNo anahtar alan olduğuna göre en fazla 1 kayıt silinecektir. Eğer 35 numaralı personel yoksa hiç kayıt silinmeyecektir.
  - Eğer kriter verilmezse tüm kayıtlar silinir.

SQLite

- SQLite, SQL komutları ile temel düzeyde veritabanı işlemlerinin yapılabildiği basit bir ilişkisel veritabanıdır.
- Kurulum gerektirmeyen yapısı nedeniyle aslında SQL ile çalışmayı destekleyen rasgele erişimli dosyalama yapısı gibi düşünülebilir.
- Kullanımı için gerekli modül (sqlite3) Python diline dahil olarak gelir.
- Çoğu ilişkisel VT'nin aksine tablolardaki sütunların veri tiplerinin statik olmaması, dinamik bir yapıda olması nedeniyle Python'daki değişkenlerin veri tiplerinin dinamik olması ile benzerlik gösterir.

connect ile SQLite'a bağlanma

- import sqlite3 ile gerekli modül programa eklendikten sonra connect yöntemi kullanılarak bir veritabanına bağlanılabilir:

```
baglanti = sqlite3.connect("data.db")
```

- Eğer Python kodunuz ile aynı dizinde "data.db" adında bir dosya yoksa yaratılır.
- Dosya adı olarak ":memory:" yazarsanız veritabanınız diskte değil hafızada yaratılır.
- Bu dosya bizim daha sonra yaratacağımız tabloları içerecektir.

Birçok ilişkisel VTYS bir sunucu hizmeti yürütür ve o hizmete bağlanmak için connect ifadesi kullanılır. Fakat SQLite bir hizmet içermediği için connect aslında burada open anlamındadır.

cursor nesnesi ve execute yöntemi

- Önceki slayttaki gibi bir bağlantı nesnesine sahip olduktan sonra, bu bağlantı ile ilişkili bir cursor nesnesi oluşturarak, oluşturulan cursor nesnesinin execute yöntemi ile SQL ifadelerini veritabanı üzerinde yürütebilirsiniz.

```
cur = baglanti.cursor()
cur.execute("CREATE TABLE …")
```

- sqlite3 ve diğer VT'ler için kullanacağımız pyodbc modülleri baglanti nesnesi üzerinde de execute yöntemi içerir, yani baglanti.execute("...") şeklinde cursor oluşturmadan da SQL ifadesi yürütülebilir.

executemany yöntemi

- Eğer elinizde bir liste içinde kayıtlar varsa bunları tek tek execute ile veritabanına eklemek yerine executemany kullanabilirsiniz.
- Bu yöntem ilk parametre olarak bir SQL ifadesi alır ve ikinci parametre olarak aldığı listenin her elemanı için bu ifadeyi çalıştırır.

```
liste = [(1, 'Altan', 'Mesut', 1234.45),
         (2, Murat’, Aslanyürek', 3222.22),
         (3, None, Aslanyürek', None)]
cur.executemany("INSERT INTO Personel
VALUES(?,?,?,?)", liste)
```

Listenin her biri tuple biçiminde olan 3 elemanı kayıtları oluşturur. Her kaydın 4 elemanı da niteliklere denk gelir. Bu nedenle VALUES içinde 4 tane ? kullandık.

commit yöntemi

- execute veya executemany ile veritabanı üzerinde kayıtları değiştirecek olan INSERT, DELETE veya UPDATE işlemlerinden birini yürütürseniz, baglanti nesnesi üzerinde commit yöntemini kullanarak kayıtlar üzerinde yaptığınız değişikliği onaylamalısınız:

```
cur.execute("INSERT INTO Personel
VALUES(4, 'Murat', 'Aslanyürek',
1234.45)")
baglanti.commit()
```

CRATE, DROP ve ALTER commit gerektirmez. Çünkü onlar kayıtları değil onların saklanacağı tablo gibi yapıların tanımını değiştirir. Değişikliklerin geri alınması istenirse rollback() yöntemi kullanılabilir. sqlite3.connect("data.db", isolation_level = None) şeklinde bağlantı oluşturulursa her DML işlemi otomatik commit edilir. Bu durumda rollback ile geri alma yapılamaz.

fetchone, fetchmany ve fetchall yöntemleri

- Veritabanı üzerinde SELECT ifadesi ile bir sorgu yürütürseniz elde edilen kayıtlar cursor nesnesinde saklanır.
- fetchone yöntemi ile sorgu sonucunda elde edilen kayıtları tek tek, fetchmany ile parametre olarak verdiğiniz sayı kadar, fetchall ile de hepsini cursor üzerinden çekersiniz (dosya nesnesindeki readline ve readlines yöntemlerinde olduğu gibi kayıt çektikçe sonraki kayıtlara ilerlersiniz)

Bu yöntemler baglanti nesnesi üzerinde bulunmaz. Bu yöntemleri kullanabilmek için cursor üzerinde execute yürütmelisiniz.

Örnekler

```
>>> cur.execute("SELECT * FROM Personel")
>>> cur.fetchone()
(1, 'Altan', 'Mesut', 1234.45)
>>> cur.execute("SELECT * FROM Personel")
>>> cur.fetchmany()
[(1, 'Altan', 'Mesut', 1234.45)]
>>> cur.fetchmany(3)
[(2, 'Murat', 'Aslanyürek', 3222.22),
 (3, 'Çağan', 'Aslanyürek', None),
 (4, 'Murat', 'Aslanyürek', 1234.45)]
>>> cur.execute("SELECT * FROM Personel")
>>> cur.fetchall()
[(1, 'Altan', 'Mesut', 1234.45),
 (2, 'Murat', 'Aslanyürek', 3222.22),
 (3, 'Çağan', 'Aslanyürek', None),
 (4, 'Murat', 'Aslanyürek', 1234.45)]
```

Argüman verilmezse tek kayıt çeker, ama liste içinde gösterir

Sorgu tekrar yürütüldüğünde cursor tekrar dolar

for döngüsü ile cursor içinde gezinme

- fetchmany ve fetchall yöntemleri çektikleri kayıtları liste olarak döndürdükleri için for döngüsü ile bu listenin elemanlarını elde edebilirsiniz:

```
for i in cur.fetchall():
    print(i)
```

- cur veya baglanti nesnelerinin execute yöntemini de direkt olarak for döngüsü ile kullanabilirsiniz:

```
for i in cur.execute("SELECT * FROM …"):
    print(i)
for i in baglanti.execute("SELECT * FROM …"):
    print(i)
```

Diğer veritabanlarına bağlanma

- Farklı veritabanlarına ODBC (Open Database Connectivity) veya o veritabanına özgü bağlantı sağlayan bir modül ile bağlanabilirsiniz. Bunun için uygun bir paket indirmelisiniz. ODBC için kullanılan paketlerden biri olan pyodbc paketi pip install ile kolayca indirilebilir. Bunun için komut isteminde aşağıdaki komutu yazın:

```
C:\Users\pc>pip install pyodbc
Collecting pyodbc
  Downloading pyodbc-4.0.23-cp36-cp36m-
win_amd64.whl (60kB)
    100% |████████████████████████████████| 61kB
445kB/s
Installing collected packages: pyodbc
Successfully installed pyodbc-4.0.23
```

Bağlantı Örnekleri

- MS Access'e bağlanmak için import pyodbc ile modülü programınıza ekledikten sonra, yine connect yöntemi kullanılır:

```
conn = pyodbc.connect(
 r"Driver={Microsoft Access Driver (*.mdb, *.accdb)};"
+
 r"Dbq=C:\Users\pc\Desktop\Data.accdb;")
```

- MS SQL Server'a bağlanmak için Windows'un ODBC ayarlarından bir bağlantı oluşturarak (DSN ismi MSSQL olsun) bu ismi kullanabilirsiniz:

```
conn = pyodbc.connect('DSN=MSSQL')
```

- Oracle'a bağlanmak için ise cx_Oracle paketini indirip import cx_Oracle ile programınıza ekleyerek kullanabilirsiniz:

```
conn = cx_Oracle.connect("murat", "abc123",
                         "localhost/orcl")
```

"murat" kullanıcı adı, "abc123" şifresi, "localhost" bağlanılacak sunucu (yani kendi bilgisayarımız) ve "orcl" o sunucu üzerindeki veritabanımızın ismidir. Farklı bilgisayardaki sunucuya bağlanmak için localhost yerine o bilgisayarın IP numarası yazılabilir.

Ödev

- SQLite üzerinde saklanacak bir telefon rehberi oluşturun.
- Rehberiniz aynı kişiye ait birçok telefon numarası saklanabilecek bir yapıda olsun.
- Programınızda bir menü olsun ve ilgili seçenekler seçildiğinde kayıtlar üzerinde arama, ekleme, silme ve güncelleme (tel numarasını değiştirme gibi) işlemleri yapılabilsin.
  - Arama yaparken iki seçenek olsun: Hem telefon numarası verilip ilgili isim getirilebilsin, hem de isim verilip telefon numarası getirilebilsin.

KAYNAKLAR

- Dr. Öğr. Üyesi Altan MESUT, Ders Notları
- Arş. Gör. Dr. Emir ÖZTÜRK, Ders Notları